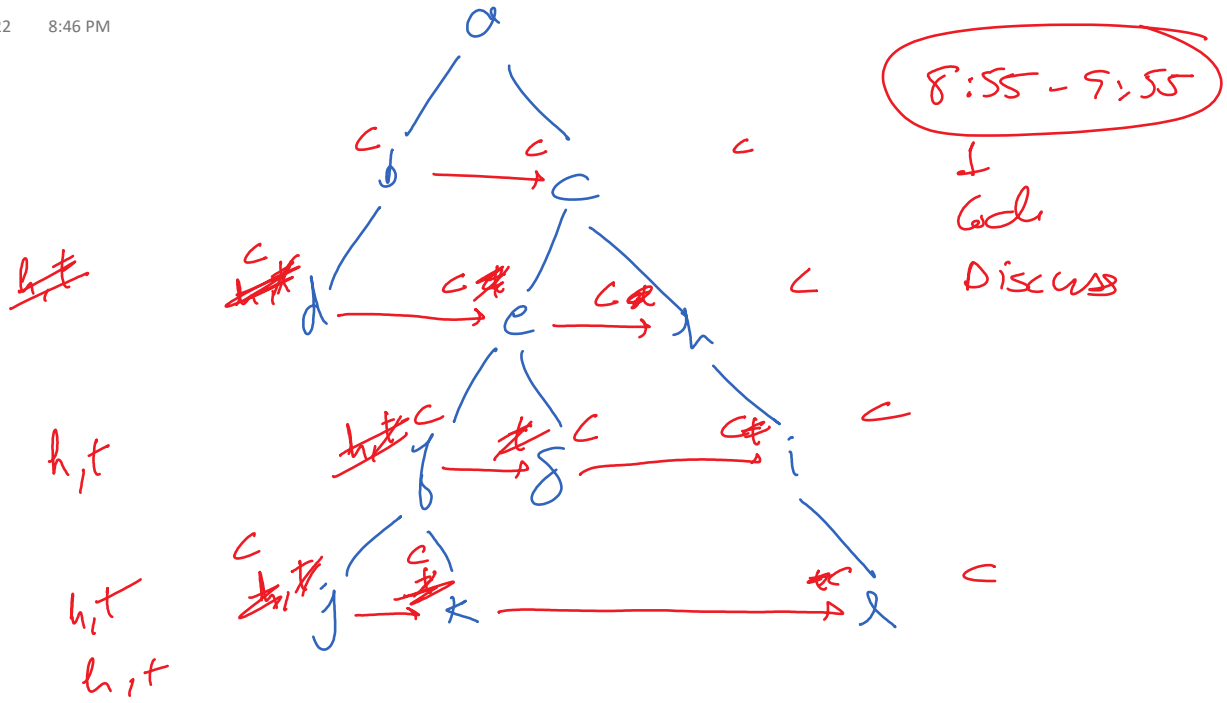
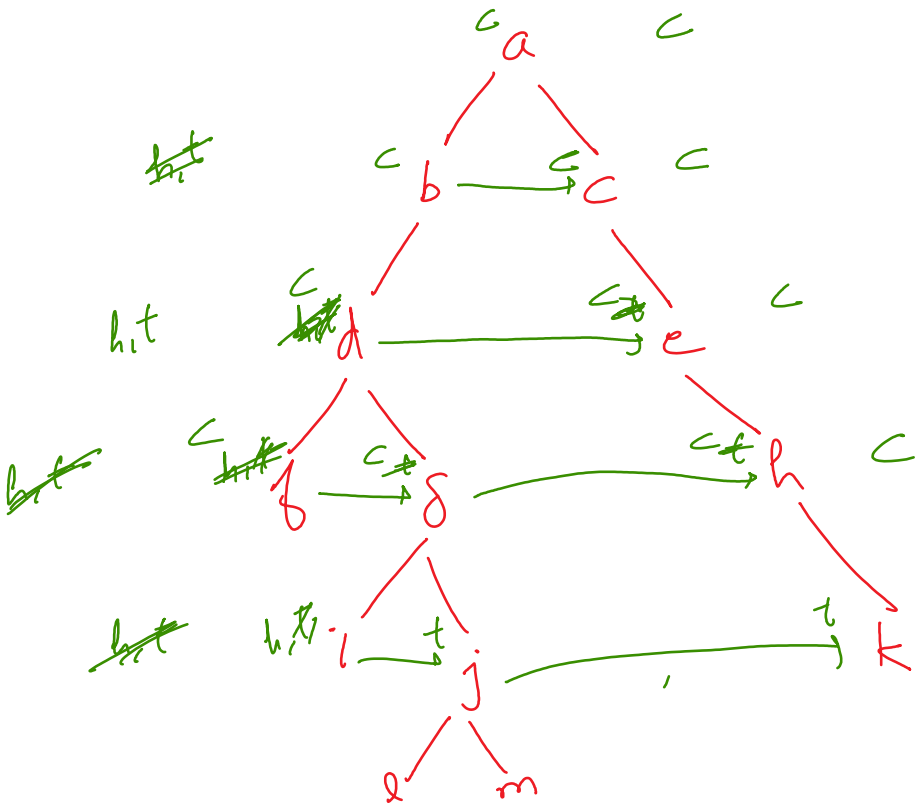
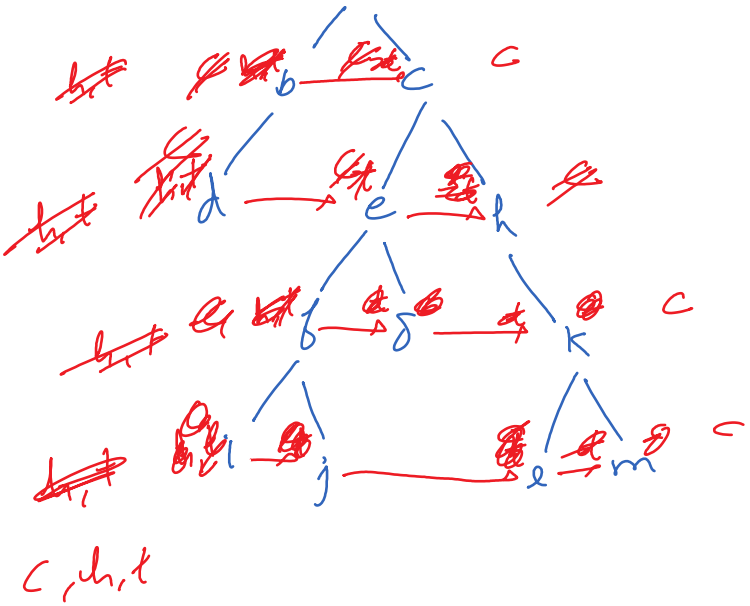
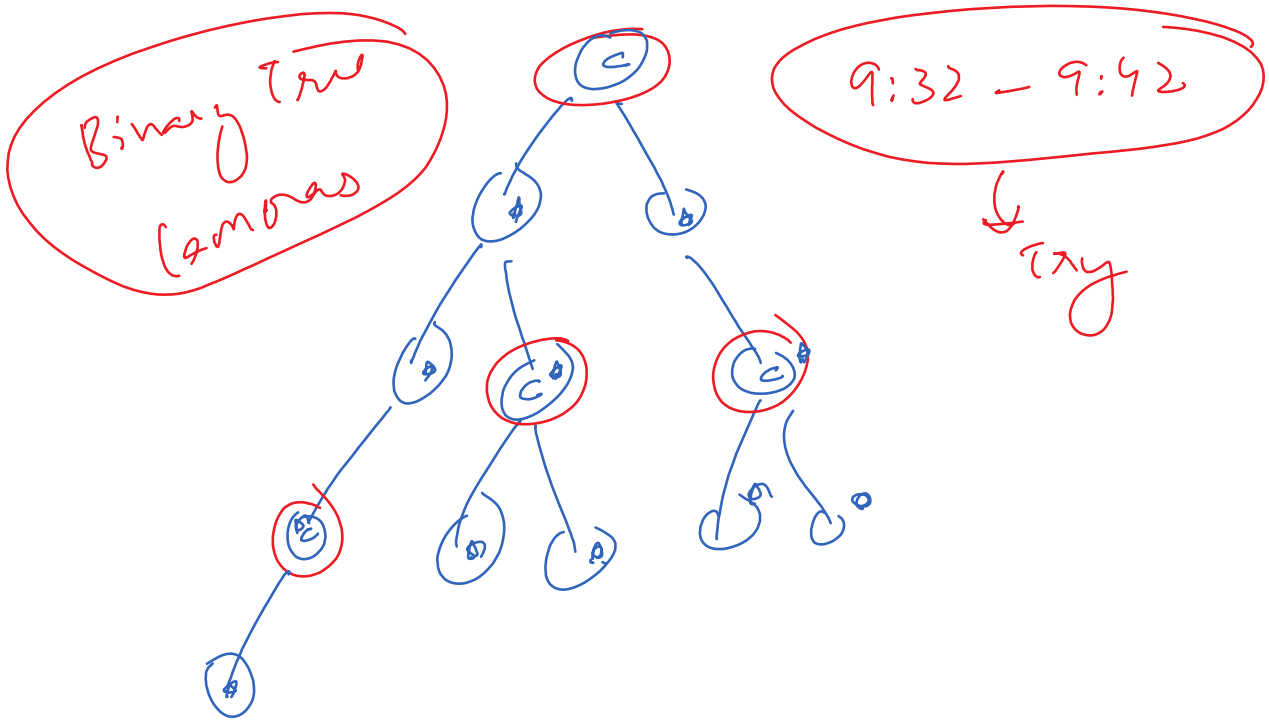


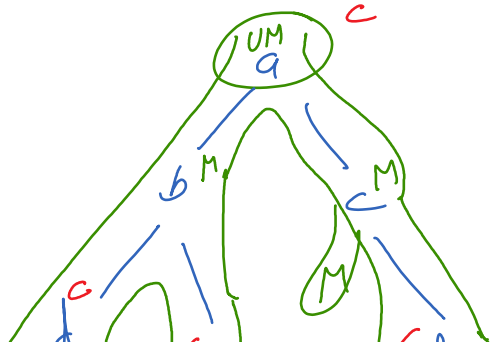


```
while(curr != null){
    // code for curr on level x, and head-
    while(curr != null){
        if(curr.left != null){
            if(head == null){
                head = curr.left;
```

```
// code for curr on level x, and head-
while(curr != null){
    if(curr.left != null){
        if(head == null){
            head = curr.left;
            tail = curr.left;
        } else {
            tail.next = curr.left;
            tail = curr.left;
        }
    }

    if(curr.right != null){
        if(head == null){
            head = curr.right;
            tail = curr.right;
        } else {
            tail.next = curr.right;
            tail = curr.right;
        }
    }

    curr = curr.next;


curr = head;
head = null;
tail = null;
```







Camera
Monitored
Not-Monitored

Monitored

Not_Monitored

l == NM || r == NM

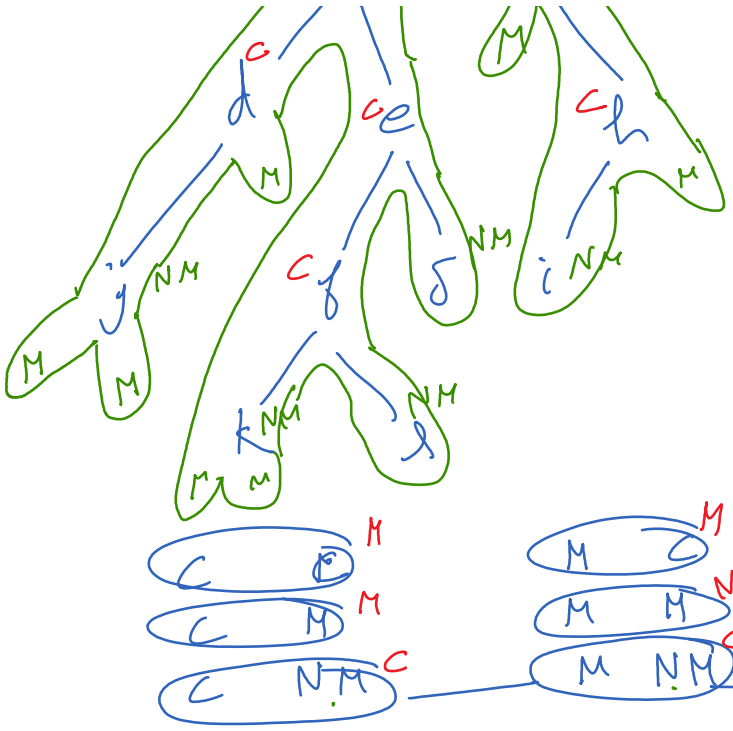
r C

l == C || r == C

r M

---

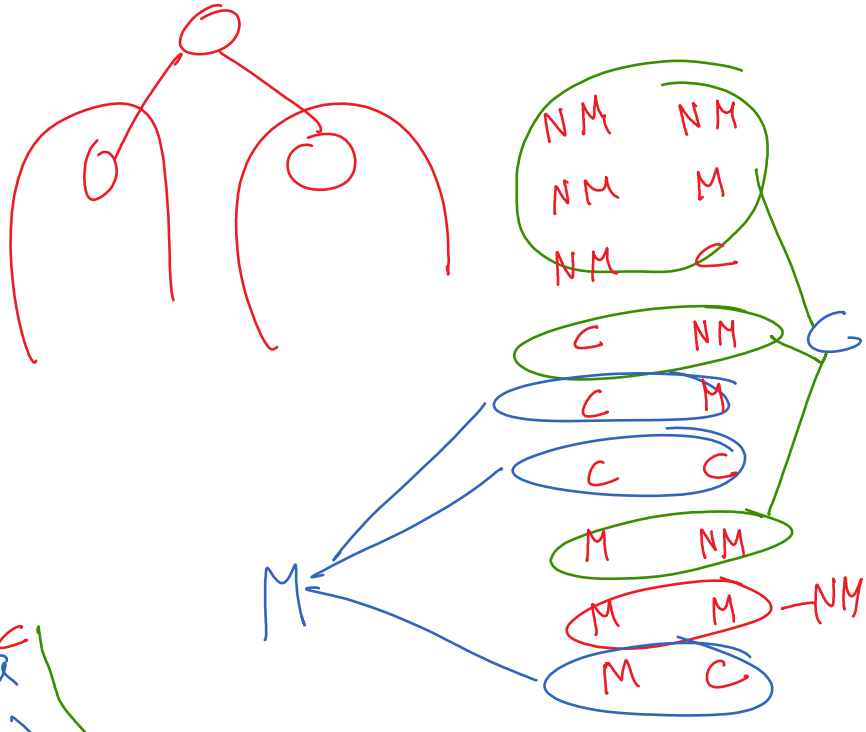
r NM

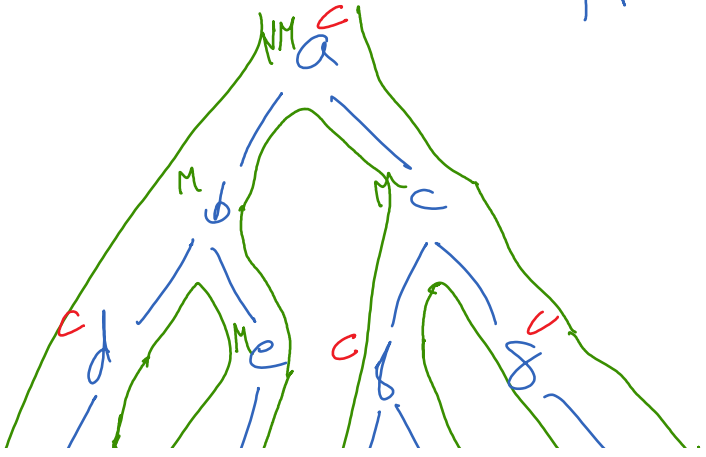


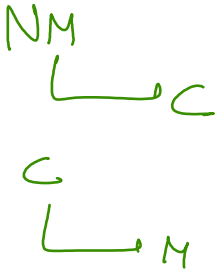
Camera

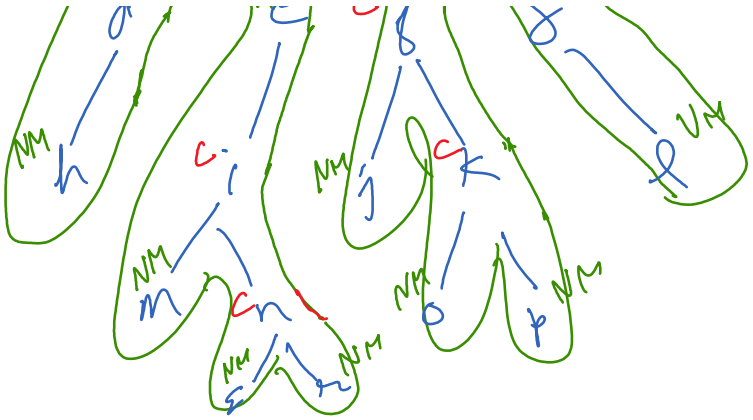
Monitored

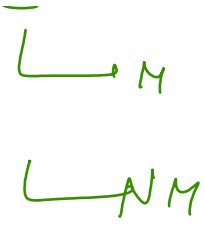
Not_Monitored

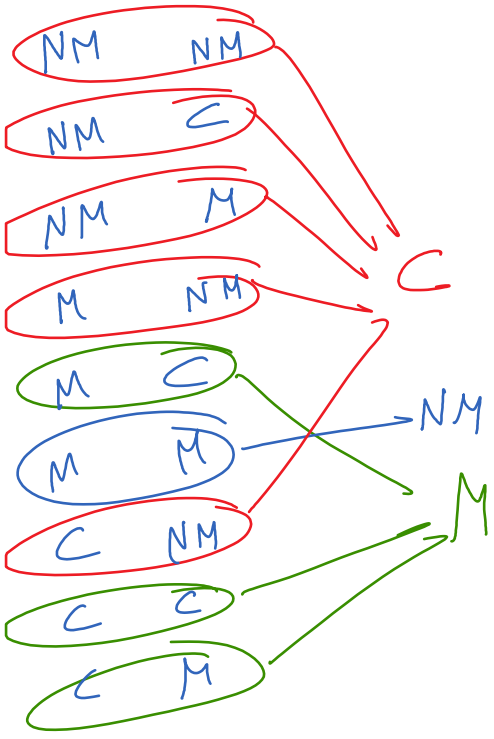


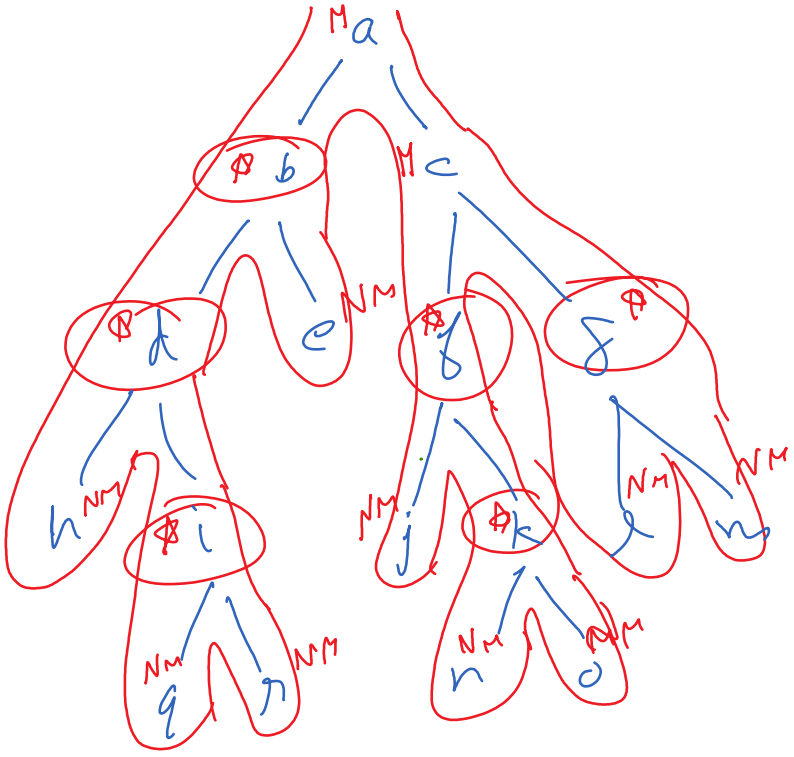


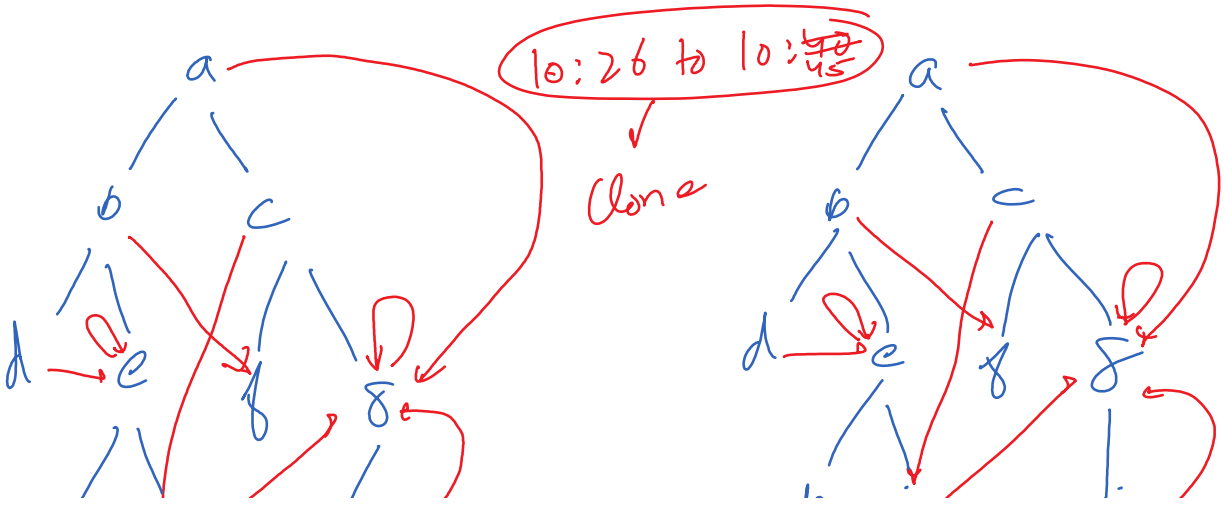


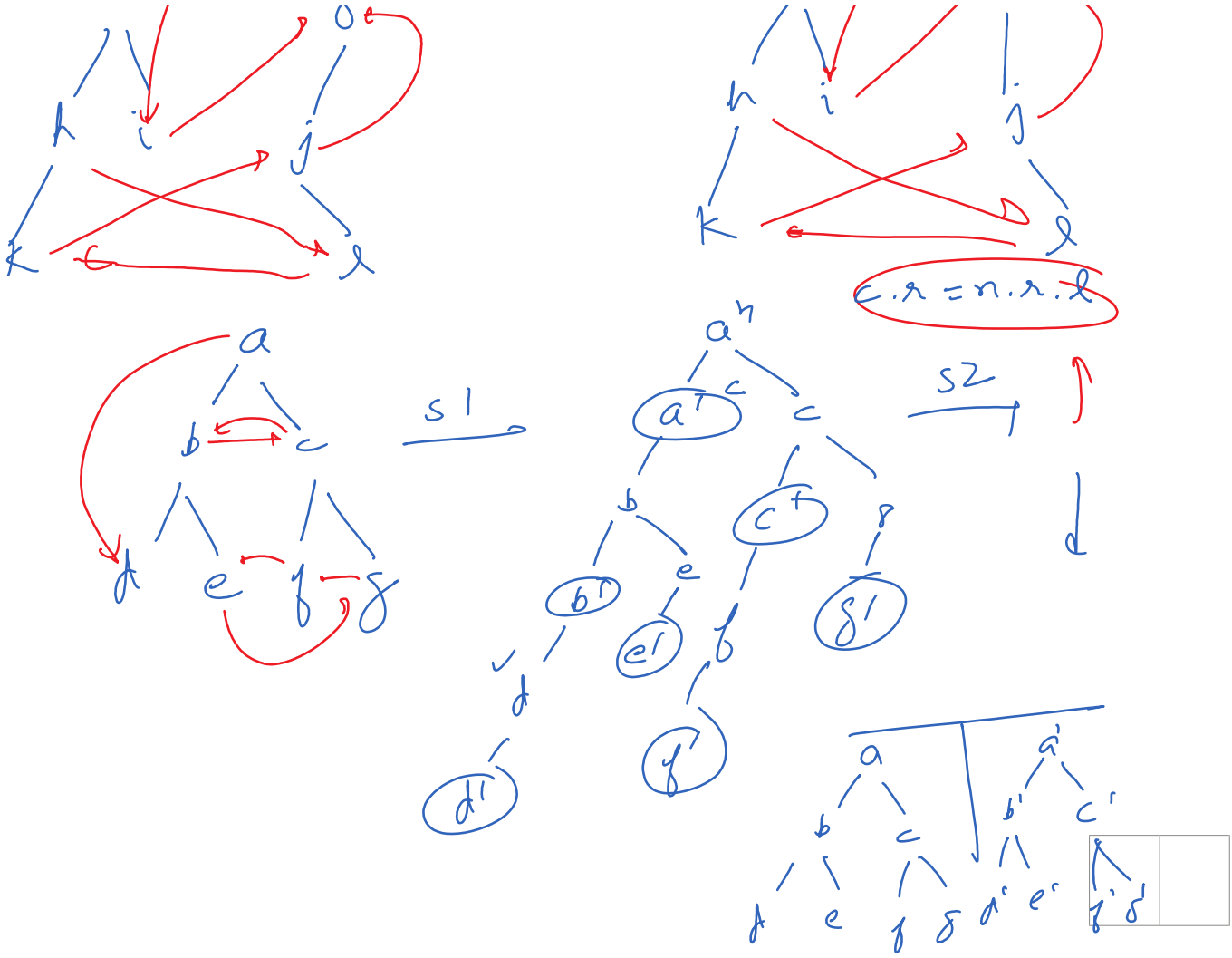


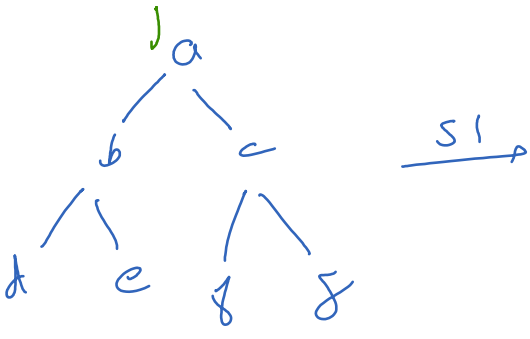




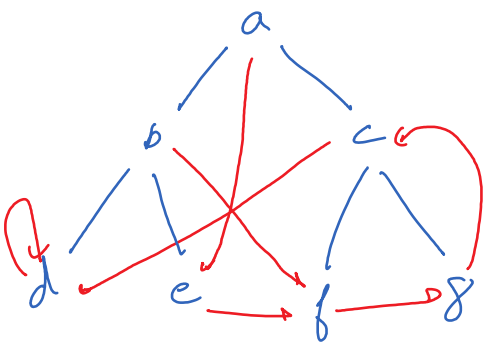


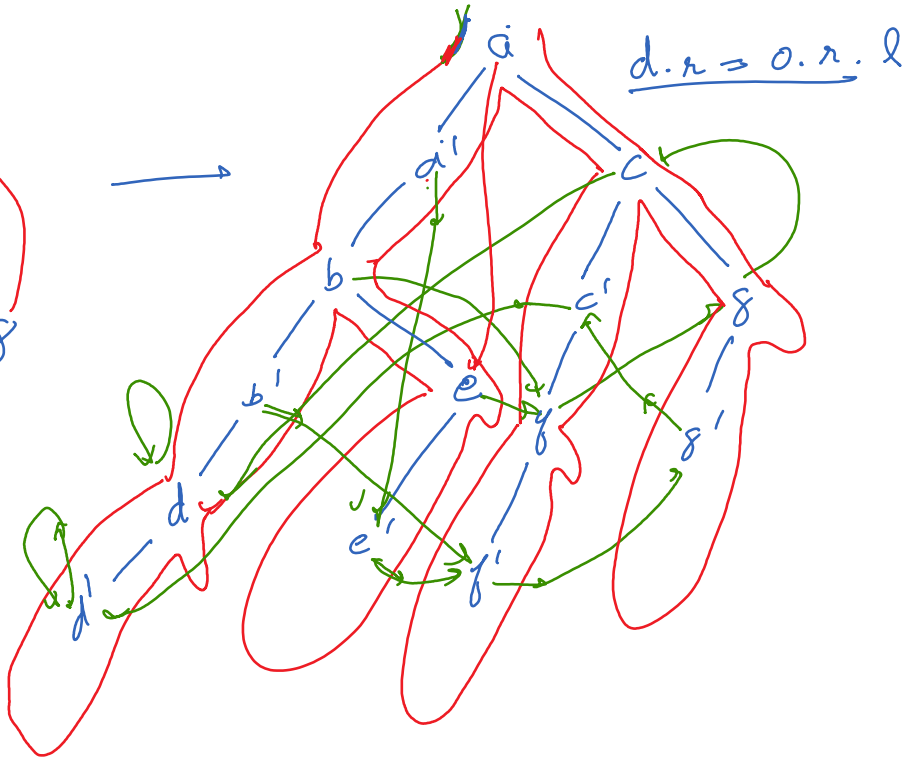

10:26 to 10:45
Clone


S1
S2
c.s = n.s.l

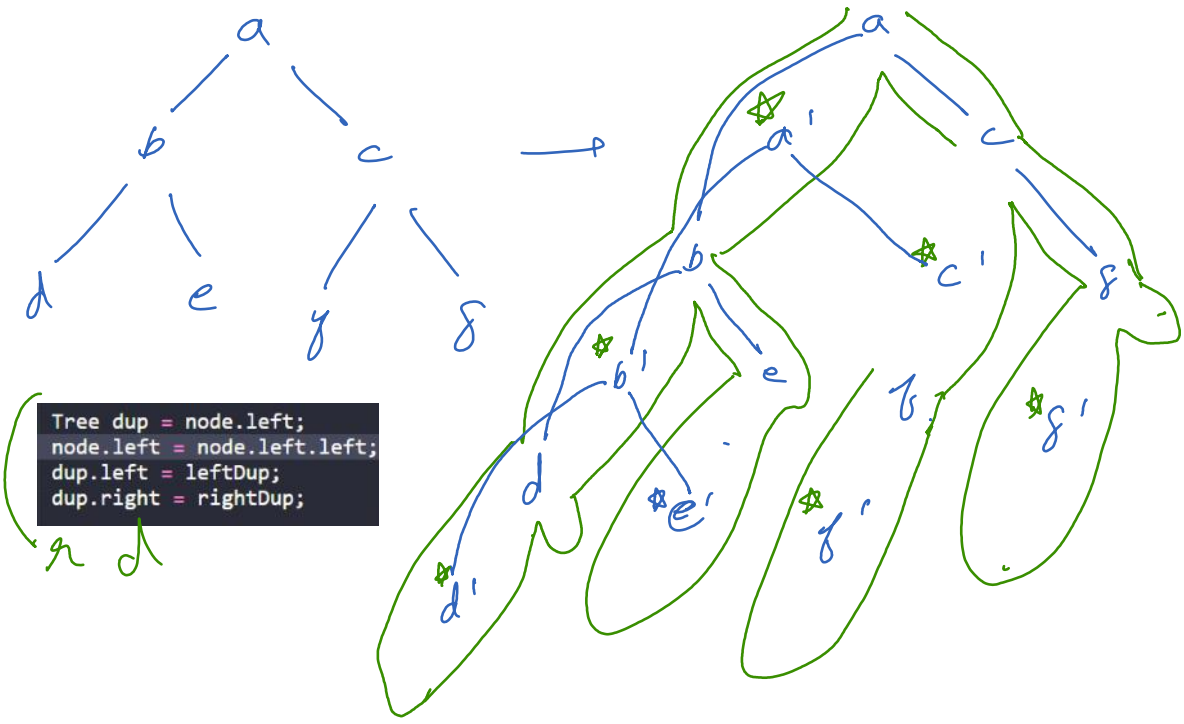

S1

```java
public void createDuplicates(Tree node){
    if(node == null){
        return;
    }

    createDuplicates(node.left);
    createDuplicates(node.right);

    Tree duplicate = new Tree(node.data);
    duplicate.left = node.left;
    duplicate.right = null;
    node.left = duplicate;
}
```

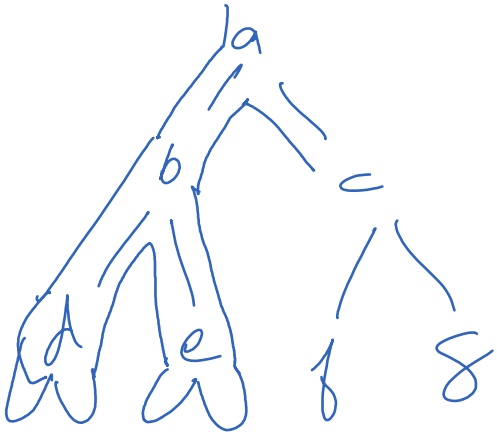


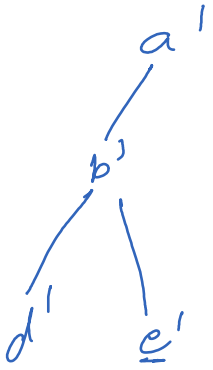



```java
public void setRandoms(Tree orig){
    if(orig == null){
        return;
    }

    setRandoms(orig.left.left);
    setRandoms(orgi.right);
    orig.left.random = orig.random.left;
}
```



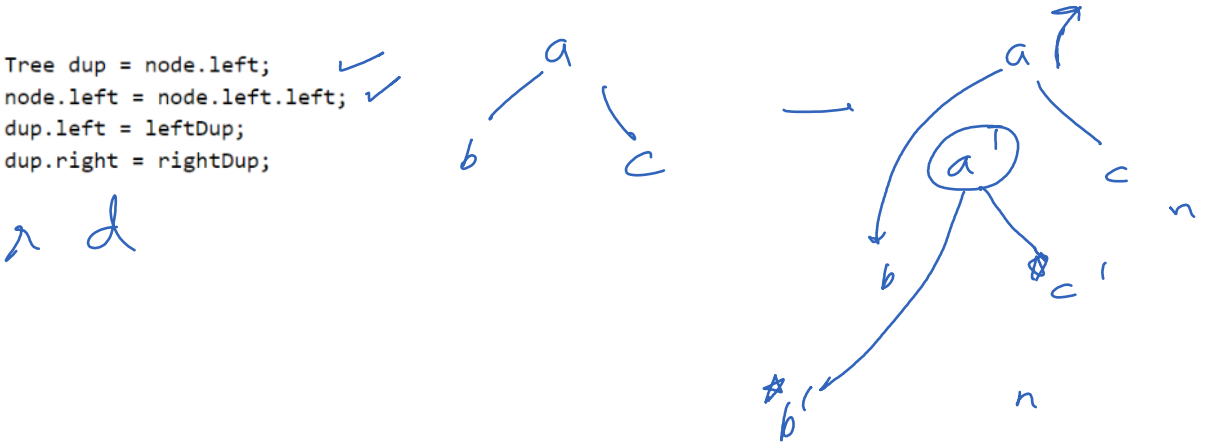
```
Tree dup = node.left;
node.left = node.left.left;
dup.left = leftDup;
dup.right = rightDup;
```

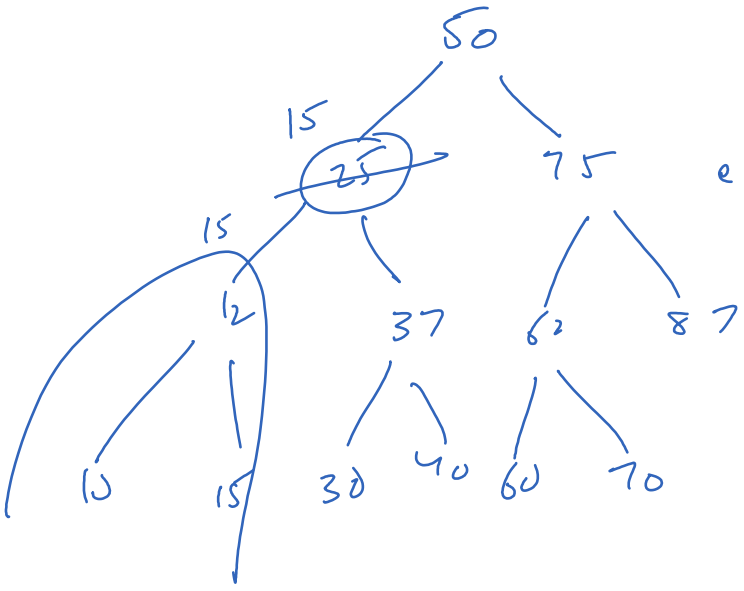




```
Node clone(Node node){
    if(temp == null){
        return null;
    }

    Node temp = new Node(node.data);
    temp.left = clone(node.left);
    temp.right = clone(node.right);

    return temp;
}
```

```
Tree dup = node.left;
node.left = node.left.left;
dup.left = leftDup;
dup.right = rightDup;
```

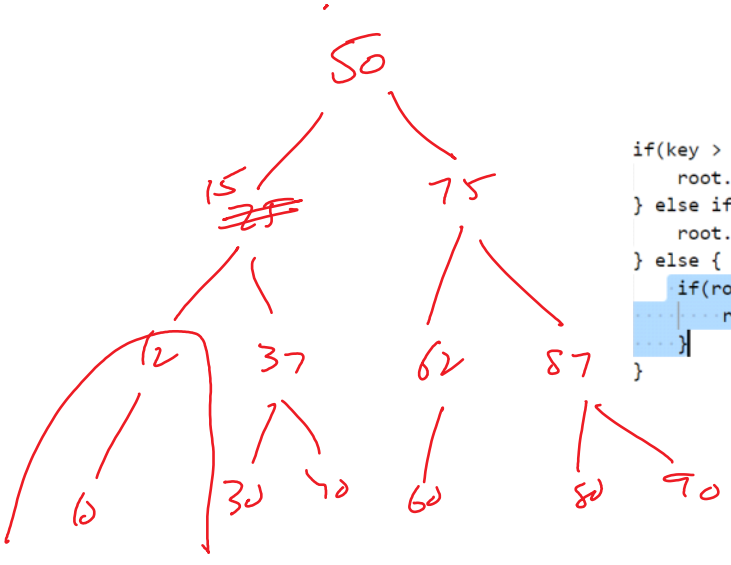




if n2d.l == null && n2d.r == null
return
else if n2d.l == null || n2d.r == null

11:50 - 12:00

Try



```
if(key > root.val){
    root.right = deleteNode(root.right, key);
} else if(key < root.val){
    root.left = deleteNode(root.left, key);
} else {
    if(root.left == null && root.right == null){
        return null;
    }
}
```